

1503.64973

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE



In Re U.S. Patent Application)
)
Applicant: Akira Tsuboi)
)
Serial No.)
)
Filed: Januaray 21, 2000)
)
For: APPARATUS AND METHOD)
FOR EXECUTING PROGRAM)
USING JUST-IN-TIME)
COMPILER SYSTEM)
Art Unit:)

I hereby certify that this paper is being deposited with the United States Postal Service as EXPRESS mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231, on December 8, 2000.

Express Label No.: EL 769180941 US

Signature: [Signature]

CLAIM FOR PRIORITY

Assistant Commissioner for Patents
Washington, DC 20231

Sir:

Applicant claims foreign priority benefits under 35 U.S.C. § 119 on the basis of the foreign application identified below:

Japanese Patent Application No. 2000-012599, filed January 21, 2000.

A certified copy of the priority document is enclosed.

Respectfully submitted,

GREER, BURNS & CRAIN, LTD.

By:

[Signature]
James K. Folker
Reg. No. 37,538

December 8, 2000
300 South Wacker Drive
Suite 2500
Chicago, IL 60606
(312) 360-0080

1503,64973
(312)360-0083

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: 2000年 1月21日

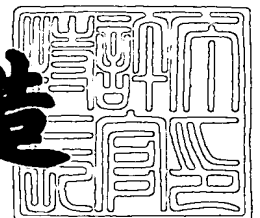
出 願 番 号
Application Number: 特願2000-012599

出 願 人
Applicant (s): 富士通株式会社

2000年 9月 8日

特許庁長官
Commissioner,
Patent Office

及 川 耕 造



出証番号 出証特2000-3071424



【書類名】 特許願

【整理番号】 9951783

【提出日】 平成12年 1月21日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/45

【発明の名称】 プログラム実行装置、プログラム実行方法、及び記録媒体

【請求項の数】 10

【発明者】

 【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

 【氏名】 坪井 晃

【特許出願人】

 【識別番号】 000005223

 【氏名又は名称】 富士通株式会社

【代理人】

 【識別番号】 100074099

 【住所又は居所】 東京都千代田区二番町8番地20 二番町ビル3F

 【弁理士】

 【氏名又は名称】 大菅 義之

 【電話番号】 03-3238-0031

【選任した代理人】

 【識別番号】 100067987

 【住所又は居所】 神奈川県横浜市鶴見区北寺尾7-25-28-503

 【弁理士】

 【氏名又は名称】 久木元 彰

 【電話番号】 045-573-3683

【手数料の表示】

 【予納台帳番号】 012542

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9705047

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 プログラム実行装置、プログラム実行方法、及び記録媒体

【特許請求の範囲】

【請求項 1】 機械語を実行する実行手段を有し、ジャスト・イン・タイム・コンパイラ方式により、原始プログラムを該実行手段で直接実行可能な機械語に翻訳して実行するプログラム実行装置であって、

前記原始プログラムに記述される関数の翻訳であって前記実行手段で実行可能な機械語を該関数ごとに記憶する、電源電圧が消失しても記憶内容が保持される記憶手段と、

前記原始プログラムを前記実行手段で実行可能な機械語に翻訳する翻訳手段と

、
前記翻訳手段により翻訳された機械語を前記記憶手段に記憶させる記憶制御手段と、

前記原始プログラムで用いられている関数の翻訳である機械語が前記記憶手段に記憶されているか否かの判定を行なう判定手段と、

前記判定手段による判定結果に応じて、前記翻訳手段の翻訳する機械語、または前記記憶手段に記憶されている機械語、のどちらかを前記実行手段に直接実行させる実行制御手段と、

を有することを特徴とするプログラム実行装置。

【請求項 2】 前記記憶手段には、前記原始プログラムで用いられることのあり得る関数の翻訳である機械語が予め記憶されていることを特徴とする請求項 1 に記載のプログラム実行装置。

【請求項 3】 前記記憶手段の記憶内容を複写して記憶する半導体メモリを更に有し、

前記実行制御手段は、前記記憶手段に記憶されている機械語を前記実行手段に実行させる代わりに、前記半導体メモリに記憶されている、該記憶手段に記憶されている記憶内容の複写である機械語を該実行手段に実行させる、

ことを特徴とする請求項 1 に記載のプログラム実行装置。

【請求項 4】 機械語を実行する実行手段を有し、ジャスト・イン・タイム

・コンパイラ方式により、原始プログラムを該実行手段で直接実行可能な機械語に翻訳して実行するプログラム実行装置であって、

前記原始プログラムに記述される関数の翻訳であって前記実行手段で実行可能な機械語を該関数ごとに記憶し、該原始プログラムの実行終了後も記憶内容を保持する記憶手段と、

前記原始プログラムを前記実行手段で実行可能な機械語に翻訳する翻訳手段と

前記翻訳手段により翻訳された機械語を、前記翻訳手段により翻訳される原始プログラムが更新された日時に対応付けて、前記記憶手段に記憶させる記憶制御手段と、

前記原始プログラムの更新された日時と、前記記憶手段に記憶されている前記機械語に対応付けられている更新日時とが一致するか否かを判定する判定手段と

前記判定手段による判定結果に応じて、前記翻訳手段の翻訳する機械語、または前記記憶手段に記憶されている機械語、のどちらかを前記実行手段に直接実行させる実行制御手段と、

を有することを特徴とするプログラム実行装置。

【請求項 5】 前記原始プログラムが格納されているプログラムファイルを読み込む読込手段を更に有し、

前記記憶制御手段は、前記プログラムファイルに示されている該プログラムファイルの更新日時を前記機械語に対応付ける該原始プログラムの更新された日時とみなして、前記記憶手段に該機械語を記憶させ、

前記判定手段は、前記プログラムファイルに示されている該プログラムファイルの更新日時と、前記機械語に対応させて前記記憶手段に記憶されている更新日時とが一致するか否かを判定する、

ことを特徴とする請求項 4 に記載のプログラム実行装置。

【請求項 6】 前記原始プログラムは、J a v a のバイト・コードで記述されていることを特徴とする請求項 1 又は 4 に記載のプログラム実行装置。

【請求項 7】 ジャスト・イン・タイム・コンパイラ方式により、原始プロ

グラムを特定の演算処理システムのプラットフォームで直接実行可能な機械語に翻訳して該機械語を実行させるプログラム実行方法であって、

前記原始プログラムの翻訳である前記機械語を、該原始プログラムに記述されている関数ごとに、電源電圧が消失しても記憶内容が保持される記憶部に記憶させ、

前記原始プログラムに記述されている関数の翻訳である前記機械語が前記記憶部に記憶されているか否かの判定を行ない、

前記判定結果に応じて、前記原始プログラムを翻訳して得る機械語、または前記記憶部に記憶されている機械語、のどちらかを特定の演算処理システムのプラットフォームで直接実行させる、

ことを特徴とするプログラム実行方法。

【請求項 8】 ジャスト・イン・タイム・コンパイラ方式により、原始プログラムを特定の演算処理システムのプラットフォームで直接実行可能な機械語に翻訳して該機械語を実行させるプログラム実行方法であって、

前記原始プログラムの翻訳である前記機械語を、該原始プログラムに記述されている関数ごとに、該機械語の翻訳前の原始プログラムが更新された日時に対応付けて記憶し、

前記原始プログラムの更新された日時と、記憶されている前記機械語に対応付けられている更新日時とが一致するか否かの判定を行ない、

前記判定結果に応じて、前記原始プログラムを翻訳して得る機械語、または記憶されている機械語、のどちらかを特定の演算処理システムのプラットフォームで直接実行させる、

ことを特徴とするプログラム実行方法。

【請求項 9】 コンピュータに実行させることによって、ジャスト・イン・タイム・コンパイラ方式により、原始プログラムを特定の演算処理システムのプラットフォームで直接実行可能な機械語に翻訳して該機械語を実行させる制御を該コンピュータに行なわせる制御プログラムを記録したコンピュータで読み取り可能な記録媒体であって、

前記原始プログラムの翻訳である前記機械語を、該原始プログラムに記述され

ている関数ごとに、電源電圧が消失しても記憶内容が保持される記憶部に記憶させるステップと、

前記原始プログラムに記述されている関数の翻訳である前記機械語が前記記憶部に記憶されているか否かの判定を行なうステップと、

前記判定結果に応じて、前記原始プログラムを翻訳して得る機械語、または前記記憶部に記憶されている機械語、のどちらかを特定の演算処理システムのプラットフォームで直接実行させるステップと、

からなる処理を該コンピュータに行なわせる制御プログラムを記録した記録媒体。

【請求項 10】 コンピュータに実行させることによって、ジャスト・イン・タイム・コンパイラ方式により、原始プログラムを特定の演算処理システムのプラットフォームで直接実行可能な機械語に翻訳して該機械語を実行させる制御を該コンピュータに行なわせる制御プログラムを記録したコンピュータで読み取り可能な記録媒体であって、

前記原始プログラムの翻訳である前記機械語を、該原始プログラムに記述されている関数ごとに、該機械語の翻訳前の原始プログラムが更新された日時に対応付けて記憶するステップと、

前記原始プログラムの更新された日時と、記憶されている前記機械語に対応付けられている更新日時とが一致するか否かの判定を行なうステップと、

前記判定結果に応じて、前記原始プログラムを翻訳して得る機械語、または記憶されている機械語、のどちらかを特定の演算処理システムのプラットフォームで直接実行させるステップと、

からなる処理を該コンピュータに行なわせる制御プログラムを記録した記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、高水準のプログラミング言語で記述されているプログラムを演算処理システムで実行させる技術に関し、特に、ジャスト・イン・タイム・コンパイ

ラ方式により、原始プログラムを特定の演算処理システムのプラットフォームで直接実行可能な機械語に翻訳して該機械語を実行させる技術に関する。

【 0 0 0 2 】

【従来の技術】

異なる演算処理システムのプラットフォーム上で同一のプログラムを動作させようとする多くの試みが以前から行なわれている。プログラミング言語 J a v a (J a v a は、サン・マイクロシステムズ・インコーポレイテッドの登録商標) はその回答のひとつである。J a v a はプラットフォーム独立を実現し、異種のプラットフォーム間での可搬性に優れている。

【 0 0 0 3 】

J a v a で記述されたプログラムのソースコードは、構文解析などの処理が施され、バイトコードと呼ばれるバイナリファイルに変換されて配布されるのが一般的である。J a v a バイトコードは演算処理システムのプラットフォームに独立である汎用的な命令コードであり、J a v a V M (バーチャル・マシン) と呼ばれる J a v a の実行系で解釈され実行されるコードである。J a v a V M に相当する環境を各々の演算処理システムが備えることによって、異なる演算処理システム上で同一の J a v a バイトコードの実行が可能となる。

【 0 0 0 4 】

以下、J a v a バイトコードを演算処理システム上で実行させる技術について説明する。

【 0 0 0 5 】

J a v a インタプリタは、J a v a バイトコードを命令毎に逐次解釈 (インタプリト) し、演算処理システムにその命令に対応する処理を行なわせるものである。J a v a インタプリタは、J a v a バイトコードで記述されたプログラムをそのまま解釈して処理できることが大きな利点ではあるが、命令解釈のための時間を要するために処理速度が遅いという短所がある。

【 0 0 0 6 】

J a v a バイトコードで記述されたプログラムの実行速度を向上させる手法として、処理させる演算処理システムで直接実行可能な機械語であるネイティブコ

ードに J a v a バイトコードを予め翻訳（コンパイル）しておく手法があり、このためのツールをネイティブ・コンパイラという。J a v a バイトコードのネイティブコード化によりプログラムの実行速度は格段に向上するが、J a v a の大きな利点である異種のプラットフォーム間での可搬性が損なわれるという問題があり、更に、プログラムの変更により J a v a バイトコードが更新される度に、演算処理システムのオペレータがネイティブコードへの翻訳作業をシステムに行なわせるように指示しなければならない煩わしさもある。

【0007】

J a v a の有する利点である異種のプラットフォーム間での可搬性を維持しつつ、J a v a インタプリタの抱える処理速度の問題を改善するためにジャスト・イン・タイム・コンパイラ（以下、「J I T コンパイラ」という）が提案されている。

【0008】

J I T コンパイラは、J a v a バイトコードを実行する際に、そのバイトコードで用いられている関数（J a v a におけるメソッド）を単位としてネイティブコードに翻訳しながら演算処理システムに逐次実行させる。ここで、翻訳されたネイティブコードをメインメモリに保持しておくようにする。J a v a バイトコードの翻訳を進めていくうちに、既にネイティブコードに翻訳されている関数が出現したときは、その関数についての再度の翻訳は行なわずに、メインメモリに保持されているネイティブコードをそのまま演算処理システムに直接実行させるようにして翻訳処理に要する処理時間を削減する。一般的にプログラム全体の実行速度を低下させる大きな影響を与えている関数は繰り返し呼び出されていることが多々あるので、この手法を用いることにより J a v a バイトコードの実行速度を向上させることができる。

【0009】

また、J I T コンパイラは、インタプリタによる解釈処理では困難な J a v a バイトコードで記述されている命令の最適化も行なうので、この最適化によっても処理速度が高速化される。しかも、ネイティブ・コンパイラでは必要となる、演算処理システムに対するオペレータによる煩雑な操作は J I T コンパイラでは

不要である。

【0010】

なお、JITコンパイラは、Javaバイトコードの実行に使用するものが特に広く普及しているが、他のプログラミング言語で記述されたプログラムや、プログラムに対して構文解析や最適化の処理が施されて得られる中間言語で記述されているプログラム（本明細書では、これらのプログラムを総称して「原始プログラム」と称することとする）を、特定のプラットフォームが直接解釈できるネイティブコードに翻訳しながらそのプラットフォーム上で実行させるように構成することも可能である。

【0011】

【発明が解決しようとする課題】

前述したように、JITコンパイラは、Javaの有する大きな利点である異種のプラットフォーム間での可搬性を維持しつつ実行速度の問題を改善することが可能であり、更にオペレータによる操作作業の負担がネイティブ・コンパイラよりも少ない利点を有している。しかしながら、原始プログラムの実行開始時にネイティブコードへの翻訳処理が必ず行なわれるため、JITコンパイラを用いて行なう原始プログラムの実行では、その原始プログラムに記述されている処理が実際に開始されるまでにある程度のタイムラグが生じてしまう問題を有していた。

【0012】

以上の問題を鑑み、JITコンパイラを用いて原始プログラムを実行させるときの実行開始時の性能を向上させることが本発明が解決しようとする課題である。

【0013】

【課題を解決するための手段】

図1は本発明の基本構成図である。本発明に係る装置であるプログラム実行装置10は、機械語を実行する実行手段16を有し、ジャスト・イン・タイム・コンパイラ方式により、原始プログラム1を実行手段16で直接実行可能な機械語に翻訳して実行する装置を前提とする。

【 0 0 1 4 】

本発明の第一の態様の装置では、プログラム実行装置 1 0 に、原始プログラム 1 に記述される関数の翻訳であって実行手段 1 6 で実行可能な機械語を該関数ごとに記憶する、電源電圧が消失しても記憶内容が保持される記憶手段 1 1 と、原始プログラム 1 を実行手段 1 6 で実行可能な機械語に翻訳する翻訳手段 1 2 と、翻訳手段 1 2 により翻訳された機械語を記憶手段 1 1 に記憶させる記憶制御手段 1 3 と、原始プログラム 1 で用いられている関数の翻訳である機械語が記憶手段 1 1 に記憶されているか否かの判定を行なう判定手段 1 4 と、判定手段 1 4 による判定結果に応じて、翻訳手段 1 2 の翻訳する機械語、または記憶手段 1 1 に記憶されている機械語、のどちらかを実行手段 1 6 に直接実行させる実行制御手段 1 5 とを有するように構成することによって前述した課題を解決する。

【 0 0 1 5 】

この構成においては、記憶手段 1 1 としては、例えばハードディスク装置やフラッシュ E E P R O M（一括消去型電氣的消去及び書き込み可能読み出し専用半導体メモリ）などを用いることができる。

【 0 0 1 6 】

上記の構成では、記憶手段 1 1 が原始プログラム 1 に記述される関数の翻訳である機械語を記憶する。原始プログラム 1 で用いられている関数の翻訳である機械語が記憶手段 1 1 に記憶されていると判定手段 1 4 が判定すれば、実行制御手段 1 5 は翻訳手段 1 2 によるその関数の翻訳を待たずに、記憶手段 1 1 に記憶されているその関数の機械語を直接実行させるように実行手段 1 6 を制御するので、プログラム実行装置 1 0 は前述した J I T コンパイラに相当する動作が行なわれる。従って、上記の構成によれば、J I T コンパイラの有する前述した利点をそのまま有している。

【 0 0 1 7 】

しかも、上記の構成における記憶手段 1 1 は、電源電圧が消失してもその記憶内容が保持される。従って、例えばオペレータが一日の仕事を終えてプログラム実行装置 1 0 の電源を切り、その翌日に仕事を再開するような場合、前日までに同一の原始プログラム 1 を一度でも実行していれば、記憶手段 1 1 に記憶されて

いる、原始プログラム 1 に記述されている関数の翻訳である機械語をその日の最初の実行から利用することができるので、プログラム実行装置 1 0 はプログラムの翻訳処理に起因する実行開始時のタイムラグを生じさせずに原始プログラム 1 を実行させることができる。

【 0 0 1 8 】

なお、上述した構成において、記憶手段 1 1 には、原始プログラム 1 で用いられることのあり得る関数の翻訳である機械語が予め記憶されているように構成しても良い。この構成では、プログラム実行装置 1 0 上での原始プログラム 1 の初めての実行の際に原始プログラム 1 に記述されている関数の翻訳である機械語が既に記憶されている場合があり、このような場合には原始プログラム 1 に記述されている関数の翻訳である機械語を記憶手段 1 1 から得ることができるので、プログラムの翻訳処理に起因する実行開始時のタイムラグを短縮することができる。

【 0 0 1 9 】

また、上述した構成において、記憶手段 1 1 の記憶内容を複写して記憶する半導体メモリを更に有し、実行制御手段 1 5 は、記憶手段 1 1 に記憶されている機械語を実行手段 1 6 に実行させる代わりに、その半導体メモリに記憶されている、記憶手段 1 1 に記憶されている記憶内容の複写である機械語を実行手段 1 6 に実行させるように構成しても良い。例えば記憶手段 1 1 としてハードディスク装置を用いている場合などでは、ハードディスク装置から機械語を読み出すよりも、そのハードディスク装置と同一の記憶内容が複写されている半導体メモリから機械語を読み出す方が、機械語の検索・読み出しが高速に行なえるので、プログラム装置 1 0 による原始プログラム 1 の翻訳・実行の処理時間を更に短縮することができる。

【 0 0 2 0 】

また、本発明の第二の態様の装置では、プログラム実行装置 1 0 に、原始プログラム 1 に記述される関数の翻訳であって実行手段 1 6 で実行可能な機械語を該関数ごとに記憶し、原始プログラム 1 の実行終了後も記憶内容を保持する記憶手段 1 1 と、原始プログラム 1 を実行手段 1 6 で実行可能な機械語に翻訳する翻訳

手段 1 2 と、翻訳手段 1 2 により翻訳された機械語を、翻訳手段 1 2 により翻訳される原始プログラム 1 が更新された日時に対応付けて、記憶手段 1 1 に記憶させる記憶制御手段 1 3 と、原始プログラム 1 の更新された日時と、記憶手段 1 1 に記憶されている前記機械語に対応付けられている更新日時とが一致するか否かを判定する判定手段 1 4 と、判定手段 1 4 による判定結果に応じて、翻訳手段 1 2 の翻訳する機械語、または記憶手段 1 1 に記憶されている機械語、のどちらかを実行手段 1 6 に直接実行させる実行制御手段 1 5 とを有するように構成することによって前述した課題を解決する。

【 0 0 2 1 】

上記の構成は、原始プログラム 1 が格納されているプログラムファイルを読み込む読込手段を更に有し、記憶制御手段 1 3 は、原始プログラム 1 が格納されているプログラムファイルに示されている該プログラムファイルの更新日時を前記機械語に対応付ける原始プログラム 1 の更新された日時とみなして、記憶手段 1 1 に該機械語を記憶させ、判定手段 1 4 は、そのプログラムファイルに示されている該プログラムファイルの更新日時と、前記機械語に対応させて記憶手段 1 1 に記憶されている更新日時とが一致するか否かを判定するように構成してもよい。

【 0 0 2 2 】

上記の構成では、前述した本発明の第一の態様の装置と同様に、J I T コンパイラの有する利点をそのまま有している。

【 0 0 2 3 】

更に、上記の構成では、原始プログラム 1 の実行終了後も記憶内容を保持しているので、同一の原始プログラム 1 を再度実行させる場合には、実行制御手段 1 5 は翻訳手段 1 2 による原始プログラム 1 の翻訳を待たずに、記憶手段 1 1 に記憶されているその関数の機械語を実行手段 1 6 に直接実行させることにより、プログラムの翻訳処理に起因する実行開始時のタイムラグを生じさせずに原始プログラム 1 を実行させることができる。ただし、この場合、後に実行される原始プログラム 1 と先に実行されたものとが、原始プログラムのバージョンアップによる修正等により、同一でないことがあり得る。そこで、記憶制御手段 1 3 が、翻

訳手段 1 2 により翻訳された機械語と、翻訳手段 1 2 により翻訳される原始プログラム 1 が更新された日時とを対応付けて記憶手段 1 1 に記憶させ、判定手段 1 4 が、原始プログラム 1 の更新された日時と、機械語に対応付けて記憶手段 1 1 に記憶されている更新日時とが一致するか否かを判定する。そしてこの判定の結果、両者が一致しないのであれば、原始プログラム 1 で用いられている関数の翻訳である機械語が記憶手段 1 1 に記憶されていても、実行制御手段 1 5 は翻訳手段 1 2 により新たに翻訳された機械語を実行手段 1 6 に実行させる。こうすることによって、原始プログラム 1 に変更が加えられても、その変更に応じて正しく原始プログラム 1 を実行させることができる。

【 0 0 2 4 】

なお、上記の構成においては、記憶手段 1 1 として、ハードディスク装置やフラッシュ E E P R O M などの、電源電圧が消失しても記憶内容が保持されるものに限らず、他の読み書き可能な記憶媒体を使用することもできる。

【 0 0 2 5 】

なお、上述した本発明の第一または第二の態様の装置において、原始プログラム 1 は、J a v a のバイト・コードで記述されていてもよい。この場合、原始プログラム 1 に記述される関数とは J a v a におけるメソッドに相当する。

【 0 0 2 6 】

また、本発明に係るプログラム実行方法は、ジャスト・イン・タイム・コンパイル方式により、原始プログラムを特定の演算処理システムのプラットフォームで直接実行可能な機械語に翻訳して該機械語を実行させる方法を前提とする。

【 0 0 2 7 】

そして、本発明の第一の態様のプログラム実行方法では、前記原始プログラムの翻訳である前記機械語を、該原始プログラムに記述されている関数ごとに、電源電圧が消失しても記憶内容が保持される記憶部に記憶させ、前記原始プログラムに記述されている関数の翻訳である前記機械語が前記記憶部に記憶されているか否かの判定を行ない、前記判定結果に応じて、前記原始プログラムを翻訳して得る機械語、または前記記憶部に記憶されている機械語、のどちらかを特定の演算処理システムのプラットフォームで直接実行させることによって前述した課題を

解決する。そして、このプログラム実行方法によっても前述した本発明の第一の態様の装置と同様な作用・効果を奏する。

【0028】

また、本発明の第二の態様のプログラム実行方法では、前記原始プログラムの翻訳である前記機械語を、該原始プログラムに記述されている関数ごとに、該機械語の翻訳前の原始プログラムが更新された日時に対応付けて記憶し、前記原始プログラムの更新された日時と、記憶されている前記機械語に対応付けられている更新日時とが一致するか否かの判定を行ない、前記判定結果に応じて、前記原始プログラムを翻訳して得る機械語、または記憶されている機械語、のどちらかを特定の演算処理システムのプラットフォームで直接実行させることによって前述した課題を解決する。そして、このプログラム実行方法によっても前述した本発明の第二の態様の装置と同様な作用・効果を奏する。

【0029】

なお、上述した本発明の各構成により行なわれる機能と同様の制御をコンピュータに行なわせる制御プログラムを記録したコンピュータ読み取り可能な記録媒体から、その制御プログラムをコンピュータに読み取らせて実行させることによって、前述した課題を解決することができ、この構成によっても前述したプログラム実行装置と同様な作用・効果を奏する。

【0030】

【発明の実施の形態】

以下、本発明の実施の形態を図面に基づいて説明する。なお、ここでは、Javaのバイトコードを翻訳して実行するプログラム実行装置において本発明を実施する例について説明する。

【0031】

図2は本発明を実施するプログラム実行装置の全体構成を示す図である。同図に示すように、このプログラム実行装置（以下、「本装置」という）20は、入力部21、CPU22、出力部23、I/F部24、半導体メモリ25、ハードディスク装置26を有し、バス27を介して相互に接続されている。

【0032】



入力部 2 1 は、キーボード装置や、マウス等のポインティングデバイスなどを有し、本装置のオペレータからの各種指示の入力を取得する。また、フロッピーディスクや光磁気ディスク、磁気テープなどの記録媒体からデータを読み出すためのデータ読み出し装置を有するようにしてもよい。

【 0 0 3 3 】

C P U 2 2 は半導体メモリ 2 5 に記憶された制御プログラムに従って本装置 2 0 全体の動作を制御する中央演算処理装置であり、機械語を直接実行する。

【 0 0 3 4 】

出力部 2 3 は、ディスプレイ装置やプリンタなどを有し、本装置 2 0 で行なった処理の結果等をオペレータに提示するものである。

【 0 0 3 5 】

I / F 部 2 4 は、本装置 2 0 を他の装置やネットワークに接続するためのインターフェースのための処理を司るものである。

【 0 0 3 6 】

半導体メモリ 2 5 は、本装置 2 0 全体の制御処理を C P U 2 2 に行なわせるための制御プログラムを予め記憶しておいたり、ハードディスク装置 2 6 の記憶内容を複写して記憶したり、あるいは C P U 2 2 により実行される処理のためのワークエリアとして用いられるものであり、R O M (リード・オンリ・メモリ) 2 5 - 1 及び R A M (ランダム・アクセス・メモリ) 2 5 - 2 を有している。

【 0 0 3 7 】

ハードディスク装置 2 6 は、本装置 2 0 へ供給される電源電圧が消失しても記憶内容が保持されるデータ記憶装置である。

【 0 0 3 8 】

次に図 3 について説明する。図 3 は、本装置 2 0 における翻訳・実行処理の全体の流れを図で示したものである。

【 0 0 3 9 】

プログラムソースモジュール 4 1 は、本装置 2 0 に翻訳・実行を行なわせるソフトウェアモジュールであり、本実施の形態では J a v a のバイトコードによって記述されている。プログラムソースモジュール 4 1 は、前述した記録媒体に格

納されているプログラムファイルとして提供されたものを入力部 2 1 のデータ読み取り装置で読み取ることにより、あるいは、他の装置やネットワークからプログラムファイルとして送付されたものを I / F 部 2 4 が受け取ることにより、本装置 2 0 に取り込まれる。

【 0 0 4 0 】

本装置 2 0 に入力されたプログラムソースモジュール 4 1 は、J I T コンパイラ 3 0 により翻訳・実行される。J I T コンパイラ 3 0 は、C P U 2 2 が本装置 2 0 全体の制御プログラムを実行することにより実現される。

【 0 0 4 1 】

J I T コンパイラ 3 0 によって行なわれる処理の内容を機能別に分けると、プログラムロード部 3 1、高速化判定部 3 2、コンパイル（翻訳）部 3 3、ファイルロード部 3 4、プログラム実行部 3 5 の各機能ブロックに大別される。

【 0 0 4 2 】

プログラムロード部 3 1 は、本装置 2 0 に取り込まれたプログラムファイルからプログラムソースモジュール 4 1 を取り出して、R A M 2 5 - 2 における C P U 2 2 がワークメモリとして使用している領域（ワークエリア）に格納する。

【 0 0 4 3 】

高速化判定部 3 2 は、プログラムソースモジュール 4 1 の構文分析を行ない、そこに記述されているメソッドの定義の翻訳に相当するネイティブコード（コンパイル済み格納モジュール 4 3）が、ハードディスク装置 2 6 に記憶されているか否かを、後述する管理簿 4 2 を参照することにより判定する。本実施の形態では、管理簿 4 2 はハードディスク装置 2 6 に記憶されている。

【 0 0 4 4 】

コンパイル部 3 3 は、上述したメソッドの定義の翻訳に相当するコンパイル済み格納モジュール 4 3 がハードディスク装置 2 6 に記憶されていないと高速化判定部 3 2 が判定したときに、そのメソッドの定義の翻訳を行ない、その結果得られた C P U 2 2 で直接実行可能なネイティブコードを、R A M 2 5 - 2 における前述したワークエリアにコンパイル済み実行モジュール 4 4 として格納すると共に、コンパイル済み格納モジュール 4 3 としてハードディスク装置 2 6 に記憶さ

せる。更に、管理簿 4 2 におけるその記憶させたコンパイル済み格納モジュール 4 3 についての管理情報を更新する。

【 0 0 4 5 】

ファイルロード部 3 4 は、上述したメソッドの定義の翻訳に相当するコンパイル済み格納モジュール 4 3 がハードディスク装置 2 6 に記憶されていると高速化判定部 3 2 が判定したときに、そのコンパイル済み格納モジュール 4 3 をハードディスク装置 2 6 からロードし（読み出し）、RAM 2 5 - 2 のワークエリアにコンパイル済み実行モジュール 4 4 として格納する。

【 0 0 4 6 】

プログラム実行部 3 5 は、コンパイル部 3 3 もしくはファイルロード部 3 4 によって RAM 2 5 - 2 のワークエリアに格納されたコンパイル済み実行モジュール 4 4 を CPU 2 2 で直接実行する。

【 0 0 4 7 】

JIT コンパイラ 3 0 は、これらの高速化判定部 3 2、コンパイル（翻訳）部 3 3、ファイルロード部 3 4、プログラム実行部 3 5 の各機能ブロックで示される処理を繰り返し実行することによって、プログラムソースモジュール 4 1 を実行させる。

【 0 0 4 8 】

次に管理簿 4 2 について説明する。図 4 は管理簿 4 2 の一例を表として示す図である。管理簿 4 2 は、ハードディスク装置 2 6 に記憶されているコンパイル済み格納モジュール 4 3 についての管理情報が記録されるものである。

【 0 0 4 9 】

図 4 に示す表において、左側の欄には、ハードディスク装置 2 6 に記憶させたコンパイル済み格納モジュール 4 3 に対応する、そのコンパイル済み格納モジュール 4 3 が翻訳である翻訳前のメソッド定義のメソッド名が記録される。そして、同図に示す表の右側の欄には、左側の欄のメソッド名に対応するメソッド定義が記述されていたプログラムソースモジュール 4 1 の更新日時が記録される。この更新日時には、本実施の形態では、プログラムソースモジュール 4 1 が格納されていたプログラムファイルの管理情報として示されている更新日時を用いる。

高速化判定部 3 2 では、プログラムロード部 3 1 が R A M 2 5 - 2 のワークエリアに格納したプログラムソースモジュール 4 1 を分析する。そして、そこに記述されているメソッド定義のメソッド名が管理簿 4 2 に記録されており、且つそのプログラムソースモジュール 4 1 の更新日時が管理簿 4 2 のそのメソッド名に対応する更新日時と一致するか否かを判定する。この判定結果が真であれば、そのメソッドの定義の翻訳に相当するコンパイル済み格納モジュール 4 3 がハードディスク装置 2 6 に記憶されていると判定する。

【 0 0 5 0 】

次に図 5 について説明する。図 5 は、R O M 2 5 - 1 に格納されている、本装置 2 0 全体の制御プログラムを C P U 2 2 が実行することにより行なわれる制御処理のうち、本発明に関係する、プログラムソースモジュール 4 1 の翻訳・実行を行なう処理である J I T コンパイル処理の第一の例の処理内容をフローチャートで示した図である。以下、同図に沿って C P U 2 2 が行なう J I T コンパイル処理について説明する。

【 0 0 5 1 】

まず、C P U 2 2 は本装置 2 0 に取り込まれたプログラムファイルからプログラムソースモジュール 4 1 を取り出し、R A M 2 5 - 2 のワークエリアに格納する (S 1 0 1) 。この処理は図 3 におけるプログラムロード部 3 1 の機能に相当する。

【 0 0 5 2 】

続いて、C P U 2 2 は、R A M 2 5 - 2 のワークエリアに格納されているプログラムソースモジュール 4 1 の構文分析を行ない、そこに記述されているメソッドの定義をひとつ抽出する (S 1 0 2) 。ここで、C P U 2 2 は管理簿 4 2 を参照し (S 1 0 3) 、抽出されたメソッド定義のメソッド名が管理簿 4 2 の記録内容に含まれているか否かを判定する (S 1 0 4) 。この判定処理の結果が Y e s ならば S 1 0 5 に進み、N o ならば S 1 0 7 に進む。

【 0 0 5 3 】

抽出されたメソッド定義のメソッド名が管理簿 4 2 の記録内容に含まれているのであれば、C P U 2 2 は、プログラムソースモジュール 4 1 が格納されていた

プログラムファイルの管理情報を調べ、プログラムソースモジュール41の更新日時を取得する(S105)。そして、このプログラムソースモジュール41の更新日時と、抽出されたメソッド定義のメソッド名の記録に対応付けられて管理簿42に記録されている更新日時とが一致するか否かを判定する(S106)。この判定処理の結果がYesならばS111に進み、NoならばS107に進む。

【0054】

以上のS102からS106にかけて示されている処理は図3における高速化判定部32の機能に相当する。

【0055】

前述したS104またはS106のどちらかの判定処理でその結果がNoであったときには、CPU22は、S102に示した処理で抽出されたメソッド定義のコンパイルを実行し(S107)、その結果得られた、CPU22で直接実行可能なネイティブコードをRAM25-2のワークエリアにコンパイル済み実行モジュール44として格納し(S108)、更にそのネイティブコードをハードディスク装置26にコンパイル済み格納モジュール43として記憶させる(S109)。このネイティブコードの記憶は、プログラムソースモジュール41の翻訳・実行が終了し、CPU22がこの図5に示したJITコンパイル処理の実行を一旦終了してもハードディスク装置26に保持されている。

【0056】

その後、CPU22は管理簿42を更新し、前述したS102でコンパイルしたメソッド定義のメソッド名と、そのメソッド定義が記述されていたプログラムソースモジュール41を格納していたプログラムファイルの管理情報に示されている更新日時とを対応付けて記録し(S110)、その後はS112に進む。

【0057】

これらのS107からS110にかけて示されている処理は図3におけるコンパイル部33の機能に相当する。

【0058】

一方、前述したS106の判定処理でその結果がYesであったときには、C



P U 2 2 は、S 1 0 2 に示した処理で抽出されたメソッド定義の翻訳であるネイティブコードをハードディスク装置 2 6 から読み出し、読み出したそのネイティブコードを R A M 2 5 - 2 のワークエリアにコンパイル済み実行モジュール 4 4 として格納する (S 1 1 1)。この処理は図 3 におけるファイルロード部 3 4 の機能に相当する。

【 0 0 5 9 】

その後、C P U 2 2 は R A M 2 5 - 2 のワークエリアにコンパイル済み実行モジュール 4 4 として格納されているネイティブコードを直接実行する (S 1 1 2)。この処理は図 3 におけるプログラム実行部 3 5 の機能に相当する。

【 0 0 6 0 】

コンパイル済み実行モジュール 4 4 を実行した後は、C P U 2 2 は、R A M 2 5 - 2 のワークエリアに格納されているプログラムソースモジュール 4 1 の実行処理を終了したか否かを判定する (S 1 1 3)。この判定処理の結果が Y e s ならば今回の J I T コンパイル処理を終了する。一方、この判定処理の結果が N o ならば S 1 0 2 へ戻り、プログラムソースモジュール 4 1 に次の実行順として記述されているメソッド定義について上述した処理を繰り返す。

【 0 0 6 1 】

以上までの処理が図 5 に示されている J I T コンパイル処理である。

【 0 0 6 2 】

次に、本発明を実施するプログラム実行装置の第二の例について説明する。

【 0 0 6 3 】

この第二の例では、図 2 におけるハードディスク装置 2 6 に、プログラムソースモジュール 4 1 で記述されることがあり得るメソッド定義の翻訳であるネイティブコードを予め格納しておくようにするものである。こうすることにより、そのプログラムソースモジュール 4 1 のプログラム実行装置 1 0 上での初めての実行の際でも、プログラムソースモジュール 4 1 に記述されているメソッド定義の翻訳であるネイティブコードをハードディスク装置 2 6 から得ることのできる場合があるので、メソッドの翻訳処理に起因する実行開始時のタイムラグを短縮することができる。

【 0 0 6 4 】

本発明を実施するプログラム実行装置の第二の例の全体構成は図 2 に示したものと同様であるが、図 2 に示す構成要素のうち、ハードディスク装置 2 6 の記憶内容が前述した第一の例と異なっている。図 6 は、本発明を実施するプログラム実行装置の第二の例におけるハードディスク装置の記憶内容を示す図であり、ハードディスク装置 2 6 には、第一の例と同様のコンパイル済み格納モジュール 4 3 が記憶されるのに加え、プログラムソースモジュール 4 1 で記述されることがあり得るメソッド定義の翻訳であるコンパイル済み標準モジュール 5 1 が予め記憶されている。このような、その翻訳であるネイティブコードがコンパイル済み標準モジュール 5 1 としてハードディスク装置 2 6 に記憶されるメソッド定義としては、例えば、J a v a の標準のクラスライブラリである j a v a パッケージに含まれるクラスに属するメソッド定義などの、任意の J a v a 実行系で使用できることが保証されているメソッド定義が適用可能である。

【 0 0 6 5 】

次に図 7 について説明する。図 7 は本発明を実施するプログラム実行装置の第二の例における管理簿 4 2 の例を表で示したものであり、ハードディスク装置 2 6 の記憶内容が図 6 に示した内容である場合に対応するものである。図 7 を、前述した本発明の第一の実施例における管理簿 4 2 の記録内容を示した図 4 と比較すると分かるように、図 7 においては、コンパイル済み格納モジュール 4 3 についての管理情報が管理簿 4 2 に記録される他に、コンパイル済み標準モジュール 5 1 についての管理情報が管理簿に予め記録されている。コンパイル済み標準モジュール 5 1 についての管理情報としては、そのコンパイル済み標準モジュール 5 1 が翻訳である翻訳前のメソッド定義のメソッド名が管理簿 4 2 に記録されている。

【 0 0 6 6 】

次に、本発明を実施するプログラム実行装置の第二の例における C P U 2 2 により実行される J I T コンパイル処理の処理内容について、図 8 に示すフローチャートを参照しながら説明する。

【 0 0 6 7 】

まず、図 8 における S 2 0 1 から S 2 0 4 にかけて示されている処理は、図 5 にフローチャートで示した J I T コンパイル処理の第一の例における S 1 0 1 から S 1 0 4 にかけて示した処理と全く同一のものであり、その説明は省略する。

【 0 0 6 8 】

S 2 0 5 において、C P U 2 2 は、S 2 0 2 で抽出されたメソッド定義のメソッド名が管理簿 4 2 のコンパイル済み標準モジュール 5 1 についての管理情報として記録されているか否かを判定する。この判定処理の結果が Y e s ならば S 2 1 2 に進み、N o ならば S 2 0 6 に進む。

【 0 0 6 9 】

図 8 における S 2 0 5 以降の、S 2 0 6 から S 2 1 4 にかけて示されている処理も、図 5 にフローチャートで示した J I T コンパイル処理の第一の例における S 1 0 5 から S 1 1 3 にかけて示した処理と全く同一のものである。

【 0 0 7 0 】

次に、本発明を実施するプログラム実行装置の第三の例について説明する。

【 0 0 7 1 】

この第三の例では、プログラム実行装置に電源を投入して起動させたときに、ハードディスク装置 2 6 の記憶内容を R A M 2 5 - 2 に複写して格納する。そして、C P U 2 2 によって行なわれる J I T コンパイル処理では、ハードディスク装置 2 6 の記憶内容に基づいて翻訳・実行の処理を進めるのではなく、ハードディスク装置 2 6 の記憶内容の複写である R A M 2 5 - 2 の格納データに従ってその処理を進める。一般的に、記憶内容の読み出しは半導体メモリからの方がハードディスク装置よりも高速に行なえるので、こうすることにより、プログラム装置 1 0 によるプログラムソースモジュール 4 1 の翻訳・実行の処理時間を更に短縮することができる。

【 0 0 7 2 】

本発明を実施するプログラム実行装置の第三の例の全体構成は図 2 に第一の例として示したものと同様であるが、図 9 に示すように、R A M 2 5 - 2 の記憶領域に、C P U 2 2 による処理の実行のために一時的に使用されるワークエリア 6 1 の領域の他に、ハードディスク装置 2 6 の記憶内容が複写されて格納されるキ

キャッシュエリア 6 2 の領域を設けるようにする。そして、このキャッシュエリア 6 2 の領域は、CPU 2 2 が後述する JIT コンパイル処理の実行を終了してもその記憶内容をクリアせずに保持するようにし、次に JIT コンパイル処理を改めて実行しても、その処理においてその記憶内容が利用できるようにする。

【 0 0 7 3 】

また、管理簿 4 2 の記録内容も図 4 に示したものと同様のものでよい。但し、前述した第一の例では管理簿 4 2 はハードディスク装置 2 6 に記憶されていたが、この第三の例では、ハードディスク装置 2 6 の管理簿 4 2 についての記録内容も RAM 2 5 - 2 に複写され、CPU 2 2 は RAM 2 5 - 2 上の管理簿 4 2 を参照して処理を行なうものとする。

【 0 0 7 4 】

本発明を実施するプログラム実行装置の第三の例における、装置全体の制御処理の処理内容を図 1 0 にフローチャートで示す。この処理は、本装置 2 0 に電源を投入するとその直後に CPU 2 2 により直ちに実行される処理である。

【 0 0 7 5 】

本装置に電源が投入されると、まず、CPU 2 2 は初期化処理を実行する (S 3 0 1) 。初期化処理は CPU 2 2 自身の有する各種のレジスタを初期化したり、RAM 2 5 - 2 の初期化などを行なう処理である。

【 0 0 7 6 】

続いて、CPU 2 2 はハードディスク装置 2 6 の記憶内容を RAM 2 5 - 2 のキャッシュエリア 6 2 に複写する (S 3 0 2) 。

【 0 0 7 7 】

その後、CPU 2 2 は、本装置 2 0 の入力部 2 1 、出力部 2 3 、I / F 部 2 4 の制御等を含む各種の制御処理を実行し (S 3 0 3) 、その後に JIT コンパイル処理を実行する (S 3 0 4) 。この処理を終えた後には S 3 0 3 へ戻り、以降の処理が繰り返される。

【 0 0 7 8 】

次に図 1 1 について説明する。図 1 1 は、図 1 0 に S 3 0 4 として示されている JIT コンパイル処理の処理内容を示すフローチャートである。

【 0 0 7 9 】

まず、図 8 における S 3 1 1 から S 3 1 9 にかけて示されている処理は、図 5 にフローチャートで示した J I T コンパイル処理の第一の例における S 1 0 1 から S 1 0 9 にかけて示した処理と同様のものである。但し、S 3 1 3 において、CPU 2 2 は、RAM 2 5 - 2 のキャッシュエリア 6 2 に格納されている、ハードディスク装置 2 6 の複写である管理簿 4 2 を参照する。

【 0 0 8 0 】

S 3 2 0 において、S 3 1 7 でのコンパイルにより得られたネイティブコードを RAM 2 5 - 2 のキャッシュエリア 6 2 にも格納し、このネイティブコードを S 3 1 9 において記憶させたハードディスク装置 2 6 の記憶内容とキャッシュエリア 6 2 の記憶内容とを一致させる。

【 0 0 8 1 】

その後、CPU 2 2 は、ハードディスク装置 2 6 及びキャッシュエリア 6 2 双方の管理簿 4 2 を同様に更新し、S 3 1 2 でコンパイルしたメソッド定義のメソッド名と、そのメソッド定義が記述されていたプログラムソースモジュール 4 1 を格納していたプログラムファイルの管理情報に示されている更新日時とを対応付けて記録し (S 3 2 1)、その後は S 3 2 3 に進む。

【 0 0 8 2 】

一方、S 3 1 6 の判定処理でその結果が Y e s であったときには、CPU 2 2 は、S 3 1 2 に示した処理で抽出されたメソッド定義の翻訳であるネイティブコードを RAM 2 5 - 2 のキャッシュエリア 6 2 から読み出し、読み出したそのネイティブコードを RAM 2 5 - 2 のワークエリアにコンパイル済み実行モジュール 4 4 として格納する (S 3 2 2)。

【 0 0 8 3 】

図 1 1 における S 3 2 2 以降の、S 3 2 3 及び S 3 2 4 に示されている処理は、図 5 にフローチャートで示した J I T コンパイル処理の第一の例における S 1 1 2 及び S 1 1 3 に示した処理と全く同一のものであり、説明は省略する。

【 0 0 8 4 】

なお、前述した第二の例、すなわちハードディスク装置 2 6 に、プログラムソ

ースモジュール 4 1 で記述されることがあり得るメソッド定義の翻訳であるネイティブコードを予め格納しておくようにしたプログラム実行装置で、この第三の例のように、ハードディスク装置 2 6 の記憶内容を R A M 2 5 - 2 に複写して J I T コンパイル処理を実行することも可能である。

【 0 0 8 5 】

なお、上述した本発明の各実施形態において説明した J I T コンパイル処理を汎用的なコンピュータで実施させることも可能である。そのためには、本発明の各実施形態において説明した、図 5、図 8、あるいは図 1 1 の J I T コンパイル処理に相当する処理をコンピュータに行なわせる制御プログラムをそのコンピュータで読み取り可能な記録媒体（記憶媒体）に予め記憶させておき、その記録媒体から読み出したその制御プログラムを読み出させてそのコンピュータのメインメモリに一旦格納させた後に、そのコンピュータの有する中央処理装置に格納されたこのプログラムを読み出させて実行させるように構成すればよい。

【 0 0 8 6 】

上述した制御プログラムを格納し、且つそれをコンピュータで読み取ることの可能な記録媒体の例を図 1 2 に示す。このような記録媒体としては、例えば、コンピュータ 7 1 の本体に内蔵若しくは外付けされる半導体メモリやハードディスク装置などのメモリ 7 2、C D - R O M、D V D - R O M、M O（光磁気ディスク）、フロッピーディスクなどといった可搬型記憶媒体 7 3、あるいはコンピュータ 7 1 と回線 7 4 で接続されていてコンピュータ 7 1 がプログラムをダウンロードすることの可能なプログラムサーバ 7 5 の記憶装置 7 6 などがあるが、これらのいずれであってもよい。

【 0 0 8 7 】

【発明の効果】

以上詳細に説明したように、本発明は、ジャスト・イン・タイム・コンパイラ方式により、原始プログラムを特定の演算処理システムのプラットフォームで直接実行可能な機械語に翻訳して該機械語を実行させるときに、前記原始プログラムの翻訳である前記機械語を、該原始プログラムに記述されている関数ごとに、電源電圧が消失しても記憶内容が保持される記憶部に記憶させ、前記原始プログラ



ムに記述されている関数の翻訳である前記機械語が前記記憶部に記憶されているか否かの判定を行ない、前記判定結果に応じて、前記原始プログラムを翻訳して得る機械語、または前記記憶部に記憶されている機械語、のどちらかを特定の演算処理システムのプラットフォームで直接実行させるように構成する。そして、この構成により、電源を切った後に同一の原始プログラムを再度実行させるような場合には、プログラムの翻訳処理に起因する実行開始時のタイムラグを生じさせずに原始プログラムを実行させることができる。

【 0 0 8 8 】

あるいは、本発明は、ジャスト・イン・タイム・コンパイラ方式により、原始プログラムを特定の演算処理システムのプラットフォームで直接実行可能な機械語に翻訳して該機械語を実行させるときに、前記原始プログラムの翻訳である前記機械語を、該原始プログラムに記述されている関数ごとに、該機械語の翻訳前の原始プログラムが更新された日時に対応付けて記憶し、前記原始プログラムの更新された日時と、記憶されている前記機械語に対応付けられている更新日時とが一致するか否かの判定を行ない、前記判定結果に応じて、前記原始プログラムを翻訳して得る機械語、または記憶されている機械語、のどちらかを特定の演算処理システムのプラットフォームで直接実行させるように構成する。そして、この構成により、こうすることによって、原始プログラムに変更が加えられても、プログラムの翻訳処理に起因する実行開始時のタイムラグを生じさせずに、且つ、その変更に応じて正しく原始プログラムを実行させることができる。

【 0 0 8 9 】

以上のように、本発明のいずれの構成によっても、JITコンパイラを用いて原始プログラムを実行させるときの実行開始時の性能を向上させることができるという効果を奏する。

【図面の簡単な説明】

【図 1】

本発明の基本構成図である。

【図 2】

本発明を実施するプログラム実行装置の全体構成を示す図である。

【図 3】

本発明を実施するプログラム実行装置における翻訳・実行処理の全体の流れを示した図である。

【図 4】

管理簿の一例を表で示した図である。

【図 5】

J I T コンパイル処理の第一の例の処理内容をフローチャートで示した図である。

【図 6】

本発明を実施するプログラム実行装置の第二の例におけるハードディスク装置の記憶内容を示す図である。

【図 7】

プログラム実行装置の第二の例における管理簿の例を表で示した図である。

【図 8】

J I T コンパイル処理の第二の例の処理内容をフローチャートで示した図である。

【図 9】

本発明を実施するプログラム実行装置の第三の例における R A M の記憶内容を示す図である。

【図 1 0】

本発明を実施するプログラム実行装置の第三の例における装置全体の制御処理の処理内容を示すフローチャートである。

【図 1 1】

図 1 0 における J I T コンパイル処理の処理内容を示すフローチャートである。

【図 1 2】

記憶させた制御プログラムをコンピュータで読み取ることの可能な記録媒体の例を示す図である。

【符号の説明】

- 1 原始プログラム
- 1 0、2 0 プログラム実行装置
- 1 1 記憶手段
- 1 2 翻訳手段
- 1 3 記憶制御手段
- 1 4 判定手段
- 1 5 実行制御手段
- 1 6 実行手段
- 2 1 入力部
- 2 2 C P U
- 2 3 出力部
- 2 4 I / F 部
- 2 5 半導体メモリ
- 2 5 - 1 R O M
- 2 5 - 2 R A M
- 2 6 ハードディスク装置
- 2 7 バス
- 3 0 J I T コンパイラ
- 3 1 プログラムロード部
- 3 2 高速化判定部
- 3 3 コンパイル部
- 3 4 ファイルロード部
- 3 5 プログラム実行部
- 4 1 プログラムソースモジュール
- 4 2 管理簿
- 4 3 コンパイル済み格納モジュール
- 4 4 コンパイル済み実行モジュール
- 5 1 コンパイル済み標準モジュール
- 6 1 ワークエリア

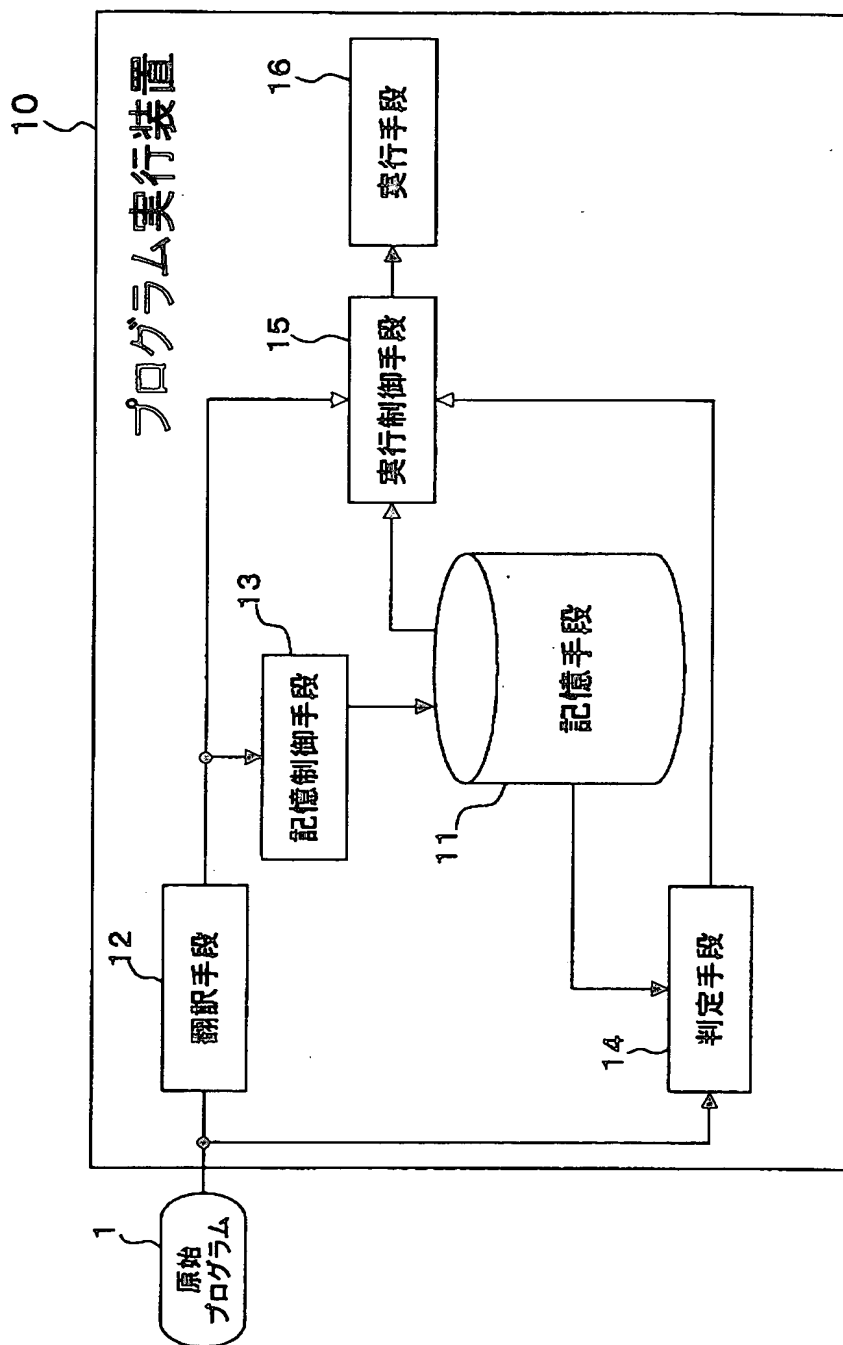
- 6 2 キャッシュエリア
- 7 1 コンピュータ
- 7 2 メモリ
- 7 3 可搬型記憶媒体
- 7 4 回線
- 7 5 プログラムサーバ
- 7 6 記憶装置

【書類名】

図面

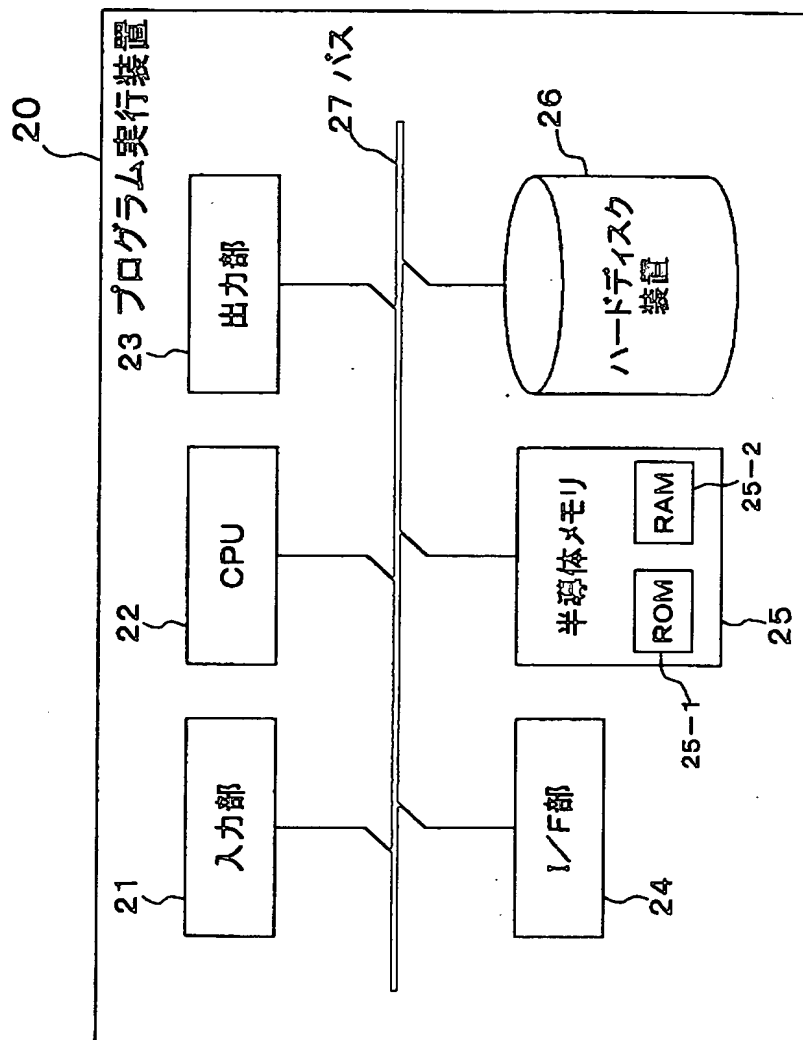
【図 1】

本発明の基本構成図



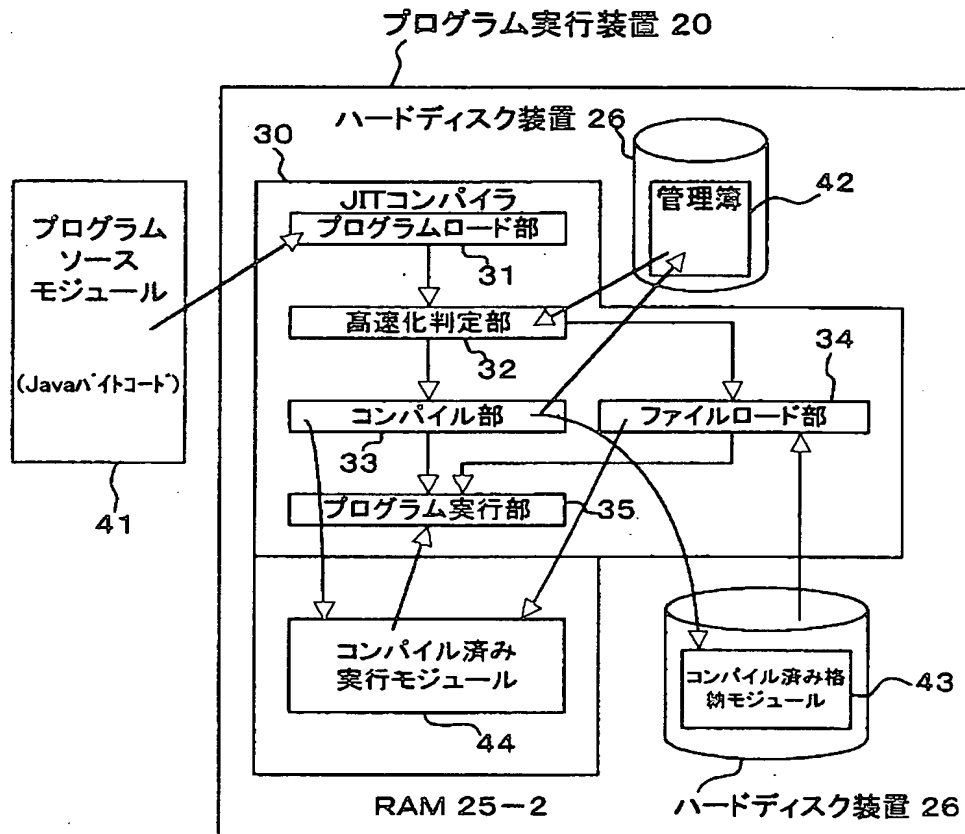
【図 2】

本発明を実施するプログラム実行装置の全体構成を示す図



【図 3】

本発明を実施するプログラム実行装置に
おける翻訳・実行処理の全体の流れを示した図



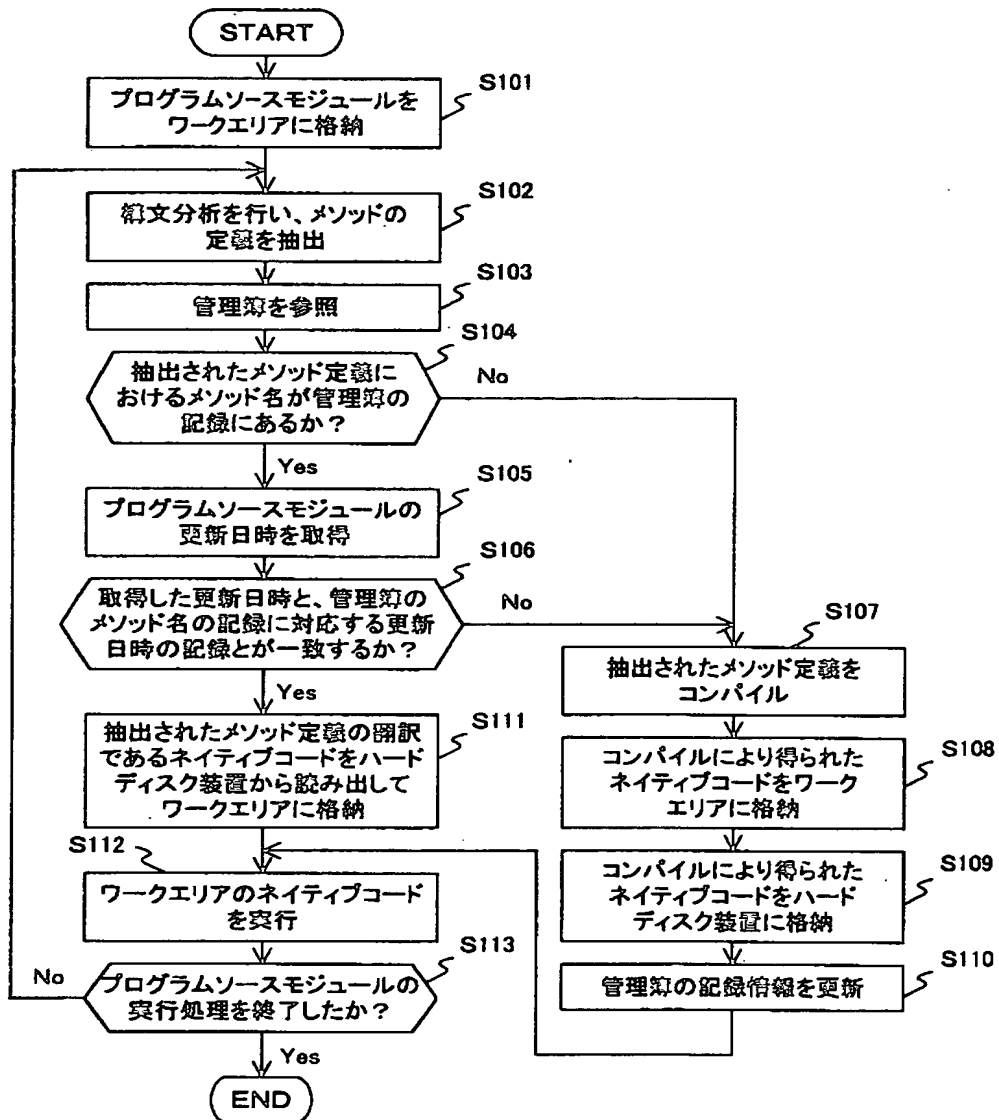
【図 4】

管理簿の一例を表で示した図

メソッド名(コンパイル対象名)	プログラムソースモジュール更新日時
open	1999. 09. 10 : 13. 25. 00
read	2000. 01. 11 : 09. 57. 34
write	2000. 01. 11 : 09. 57. 34
close	1999. 09. 10 : 13. 25. 00

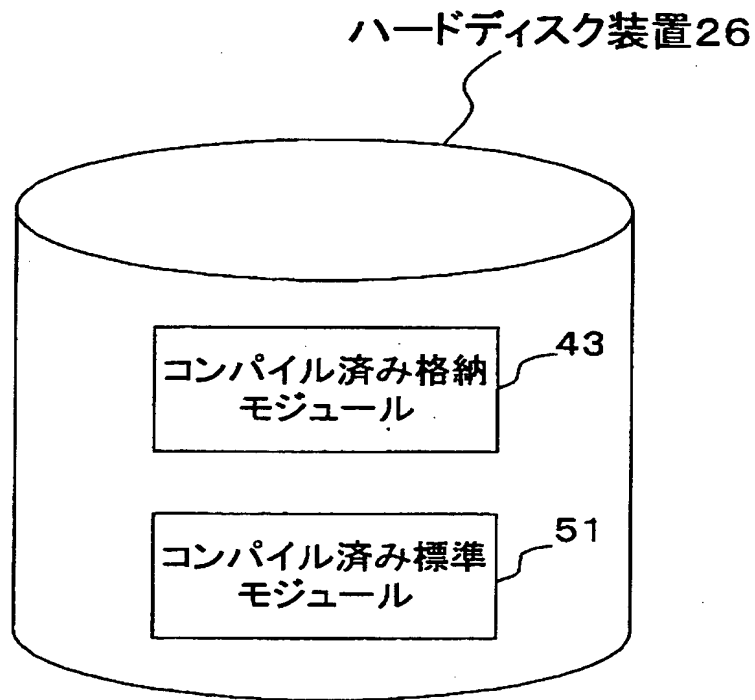
【図 5】

JITコンパイル処理の第一の例の処理内容をフローチャートで示した図



【図 6】

本発明を実施するプログラム実行装置の第二の例におけるハードディスク装置の記憶内容を示す図



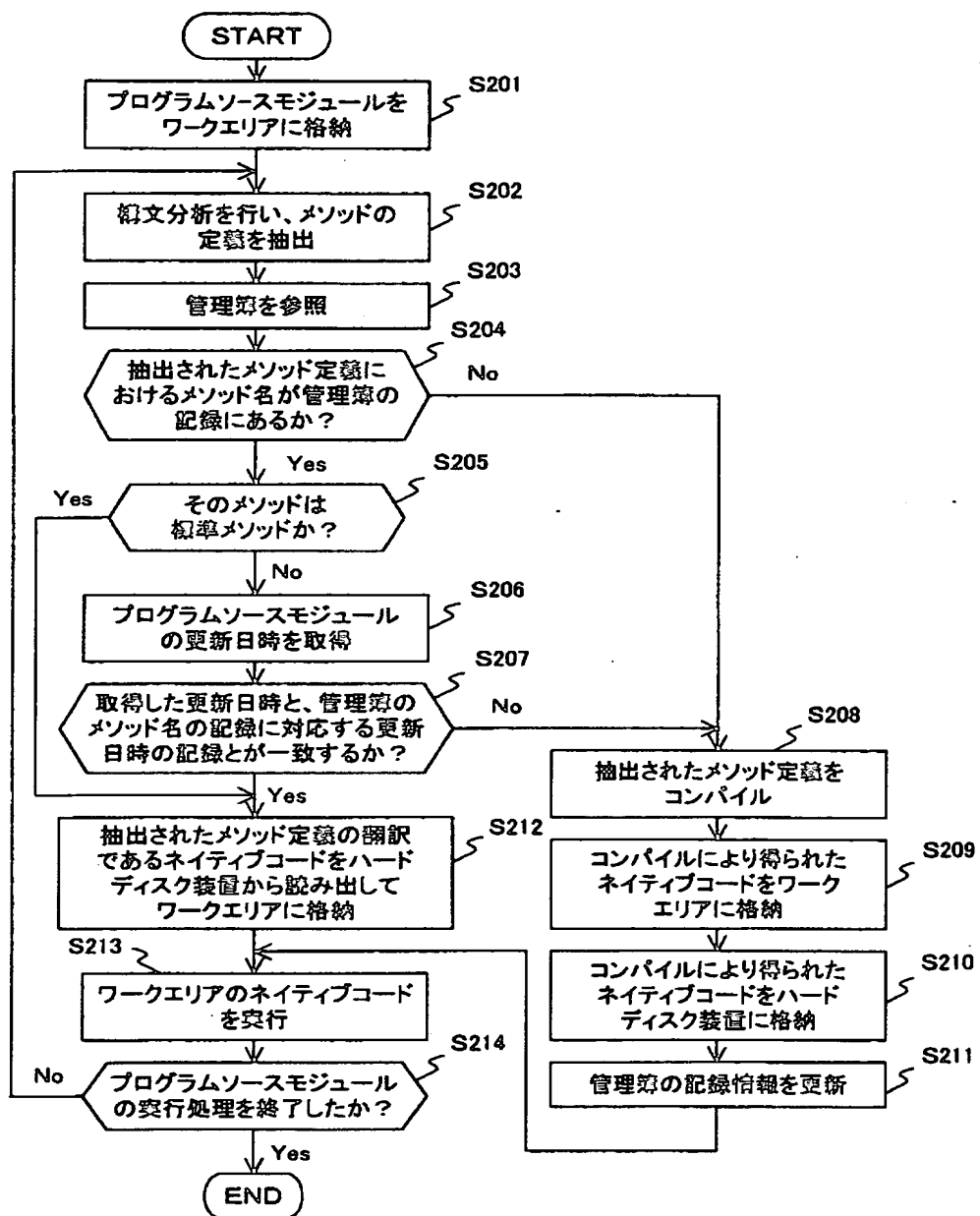
【図 7】

プログラム実行装置の第二の例における管理簿の例を表で示した図

メソッド名	プログラムソースモジュール更新日時	コンパイル済み 格納モジュールについての 管理情報	
		コンパイル済み 格納モジュールについての 管理情報	コンパイル済み 格納モジュールについての 管理情報
function 1	1999. 09. 10 : 13. 25. 00		
function 2	2000. 01. 11 : 09. 57. 34		
function 3	1999. 09. 10 : 13. 25. 00		
standard 1			
standard 2			
standard 3			

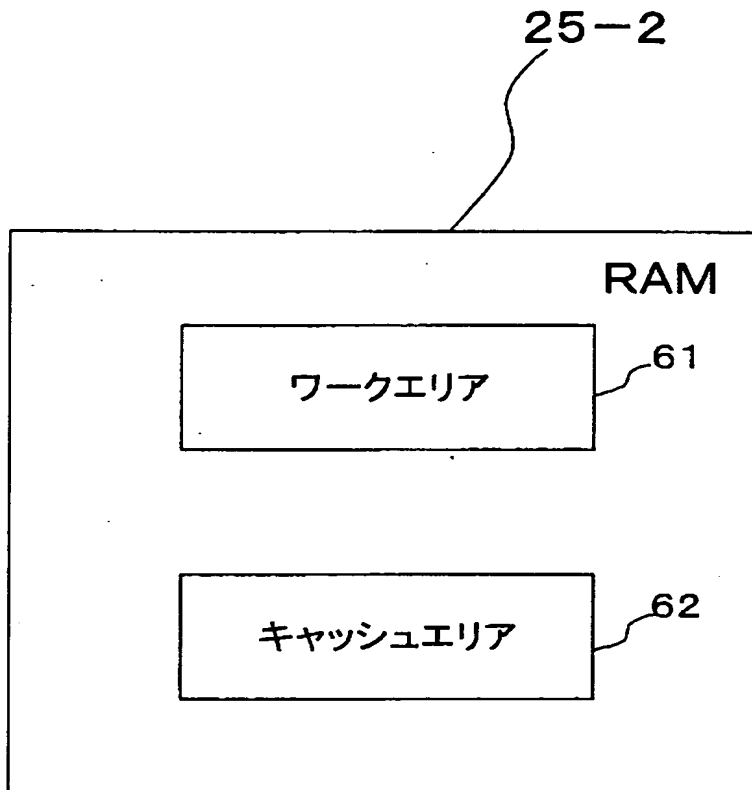
【図 8】

JITコンパイル処理の第二の例の処理内容をフローチャートで示した図



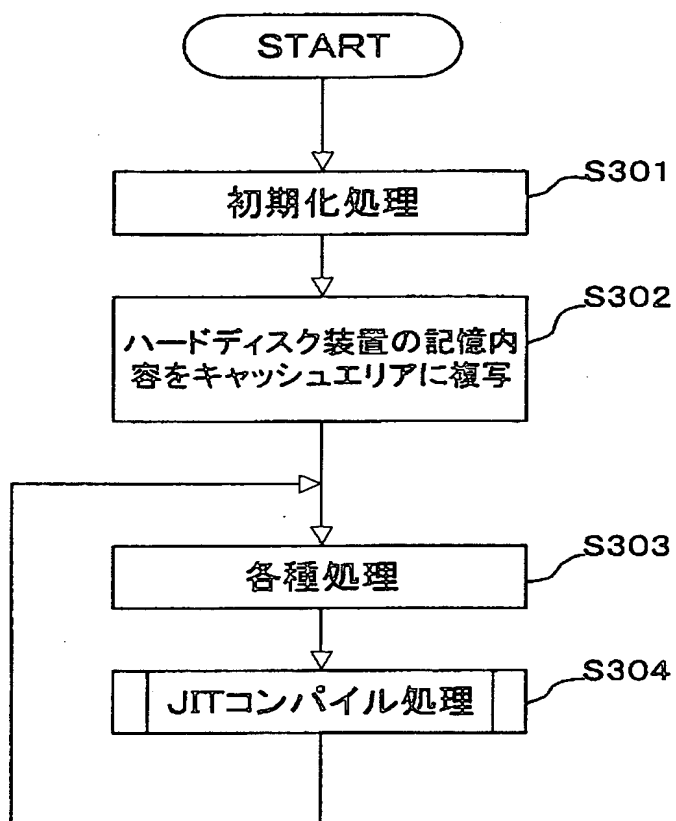
【図 9】

本発明を実施するプログラム実行装置の
第三の例におけるRAMの記憶内容を示す図



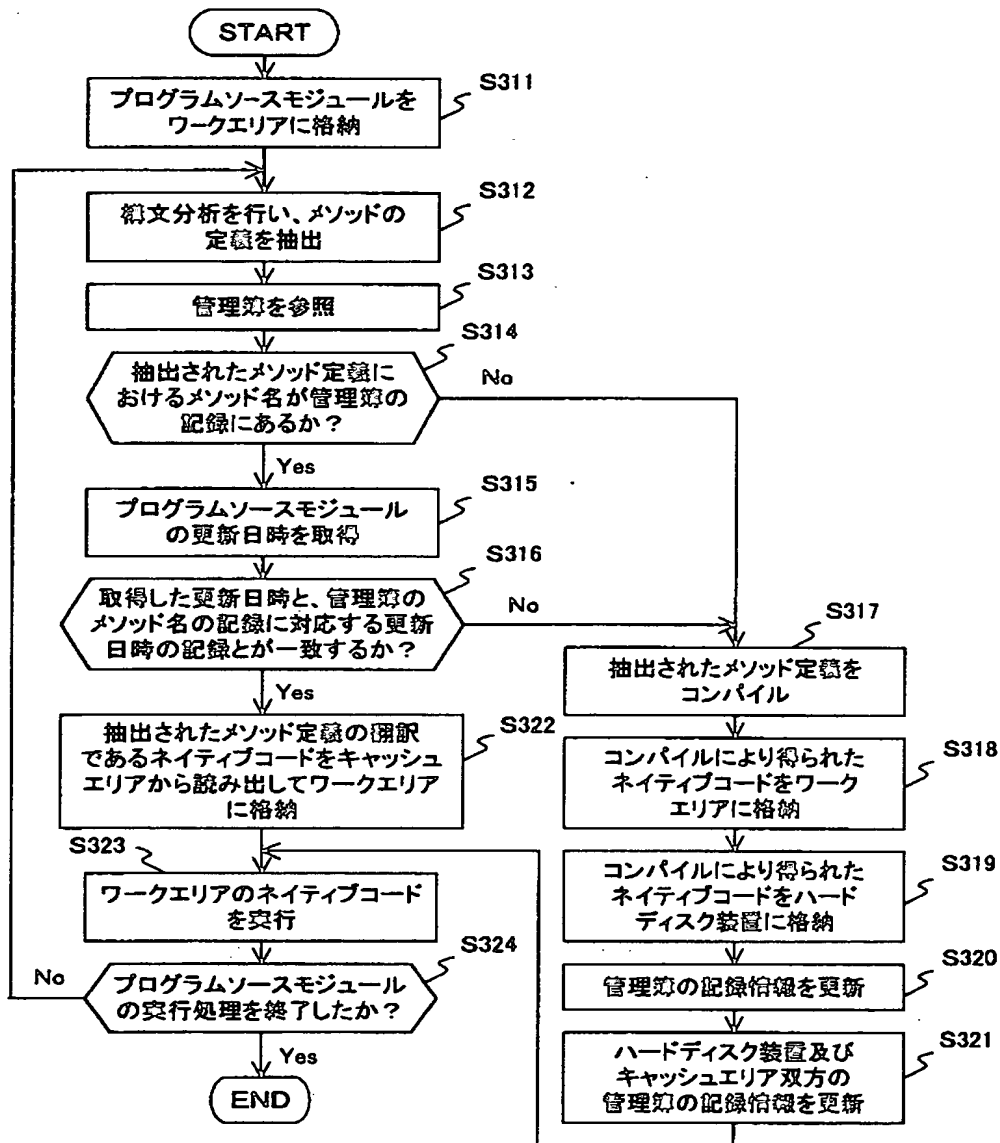
【図 1 0】

本発明を実施するプログラム実行装置の第三の例における
装置全体の制御処理の処理内容を示すフローチャート



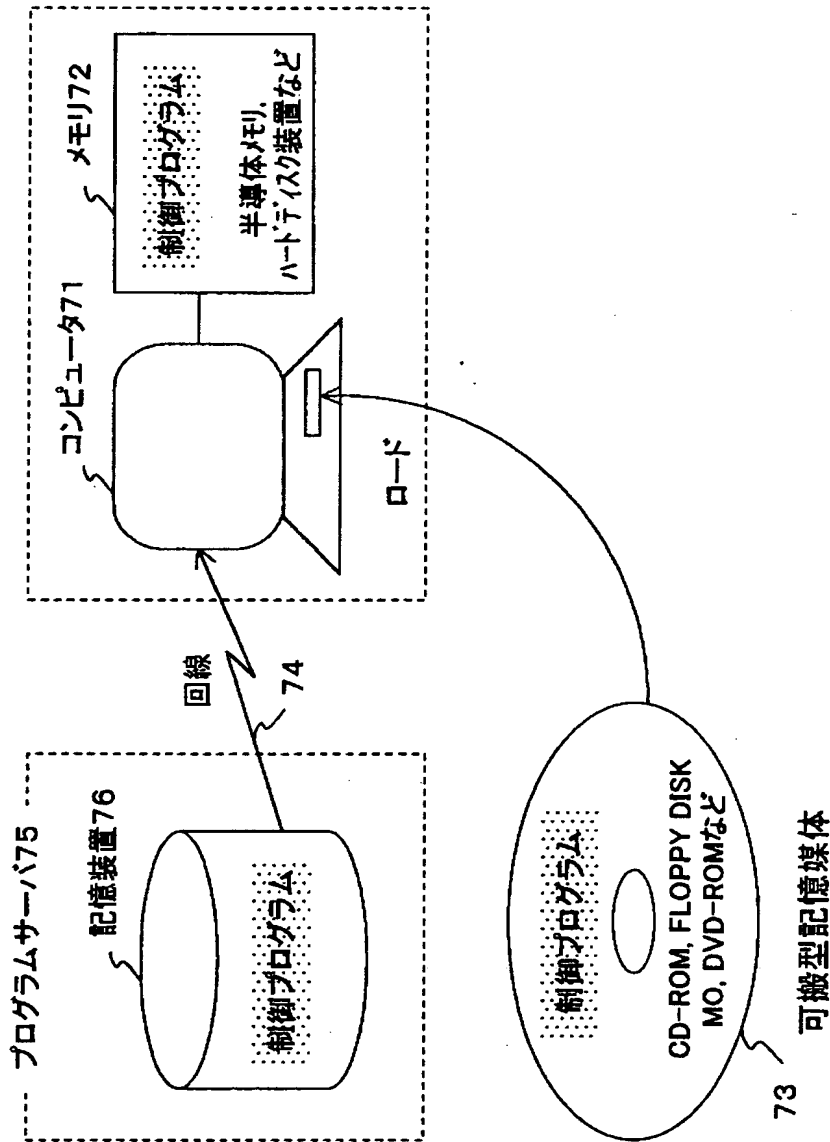
【図 11】

図 10 における JIT コンパイル処理の処理内容をフローチャート



【図 1 2】

記憶させた制御プログラムをコンピュータで
読み取ることの可能な記録媒体の例を示す図



【書類名】 要約書

【要約】

【課題】 J I T コンパイラを用いて原始プログラムを実行させるときの実行開始時の性能を向上させる。

【解決手段】 プログラム実行装置 1 0 に、原始プログラム 1 に記述される関数の翻訳であって実行手段 1 6 で実行可能な機械語を該関数ごとに記憶する、電源電圧が消失しても記憶内容が保持される記憶手段 1 1 と、原始プログラム 1 を実行手段 1 6 で実行可能な機械語に翻訳する翻訳手段 1 2 と、翻訳手段 1 2 により翻訳された機械語を記憶手段 1 1 に記憶させる記憶制御手段 1 3 と、原始プログラム 1 で用いられている関数の翻訳である機械語が記憶手段 1 1 に記憶されているか否かの判定を行なう判定手段 1 4 と、判定手段 1 4 による判定結果に応じて、翻訳手段 1 2 の翻訳する機械語、または記憶手段 1 1 に記憶されている機械語、のどちらかを実行手段 1 6 に直接実行させる実行制御手段 1 5 とを備える。

【選択図】 図 1

出 願 人 履 歴 情 報

識別番号 [000005223]

1. 変更年月日	1996年 3月26日
[変更理由]	住所変更
住 所	神奈川県川崎市中原区上小田中4丁目1番1号
氏 名	富士通株式会社